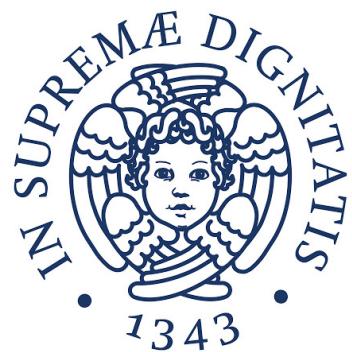


UNIVERSITÀ DI PISA



SCUOLA DI RECITAZIONE

Business Process Modeling

DATA SCIENCE & BUSINESS INFORMATICS

Authors

Marianna Abbattista 621809
Lia Trapanese 628153

2021/2022

Indice

1	Introduzione	2
2	BPMN MODEL	3
2.1	Pool Allievo	3
2.2	Pool Scuola di recitazione	4
2.3	Pool Maestro	5
3	PETRI NETS ANALISI	8
3.1	Allievo	8
3.2	Scuola di recitazione	9
3.3	Maestro	11
4	Conclusioni	12

1 Introduzione

Il problema presentato illustra il processo che coinvolge un allievo che chiede di frequentare un corso in una scuola di recitazione e quest'ultima lo mette in contatto con uno dei maestri.

Come primo step abbiamo identificato gli attori coinvolti nello scenario:

- *Allievo*
- *Scuola di recitazione*
- *Maestro*

Il processo è stato modellato utilizzando il linguaggio BPMN. Il diagramma è stato creato con l'aiuto dell'applicazione **Signavio**.

Successivamente abbiamo eseguito l'analisi della Petri Net, per definire la Soundness del processo. Ciò è stato modellato utilizzando **WoPed**, **Woflan** per la sua analisi.

Il design generale del modello è basato su una struttura a *Coreografia*. Per questo motivo nel BPMN diagram sono stati creati tre pools, uno per ogni attore. Questi interagiscono attraverso dei **message flow arrows**, i quali rappresentano il flusso di informazioni che vengono scambiate da un pool ad un altro. Gli **Artefacts** vengono utilizzati per la rappresentazione grafica degli oggetti fuori dal processo principale. Sono stati utilizzati anche dei **event based gateways**, degli **XOR join** e **XOR split**.

2 BPMN MODEL

2.1 Pool Allievo

Uno dei due pool principali è rappresentato dall'Allievo. Da qui parte tutto il nostro processo, difatti è lui che prende l'iniziativa di *"Contattare una scuola di recitazione"* e successivamente esegue il task di contatto. Una volta fatto ciò *"Richiede la lista dei corsi"* alla scuola. Da entrambi i task partirà una **message flow arrows** alla scuola, in questo momento si attiverà il pool di essa, che una volta ricevuta la richiesta, invierà la lista dei corsi. L'allievo a questo punto *"Sceglierà il corso"* e *"Richiederà il contatto con un maestro"* tramite la scuola. Riceverà da un maestro la proposta di data e luogo e *"Valuterà la proposta"*. Come si può notare qui abbiamo uno **XOR split**, difatti l'allievo può accettare la proposta o rifiutarla. Se l'accetta, *"conferma l'appuntamento"* al maestro, altrimenti *"invia una controproposta"* decidendo una nuova data e luogo (rappresentata da un **text artifact**). Se il maestro accetta la controproposta, riceverà la conferma dell'appuntamento, altrimenti un rifiuto e si ritornerà nella fase iniziale in cui il maestro fa una nuova proposta di incontro. Per rappresentare ciò abbiamo deciso di utilizzare un **event based gateways**, questo perché il processo dell'allievo continua in base ad una decisione presa esternamente (dal maestro). Il loop continua fino a quando l'allievo e il maestro non trovano un accordo, ed appena ciò avviene, si entra nel task *"Inizia il corso"*, seguito dall'invio all'allievo del testo da preparare per la lezione da parte del maestro.

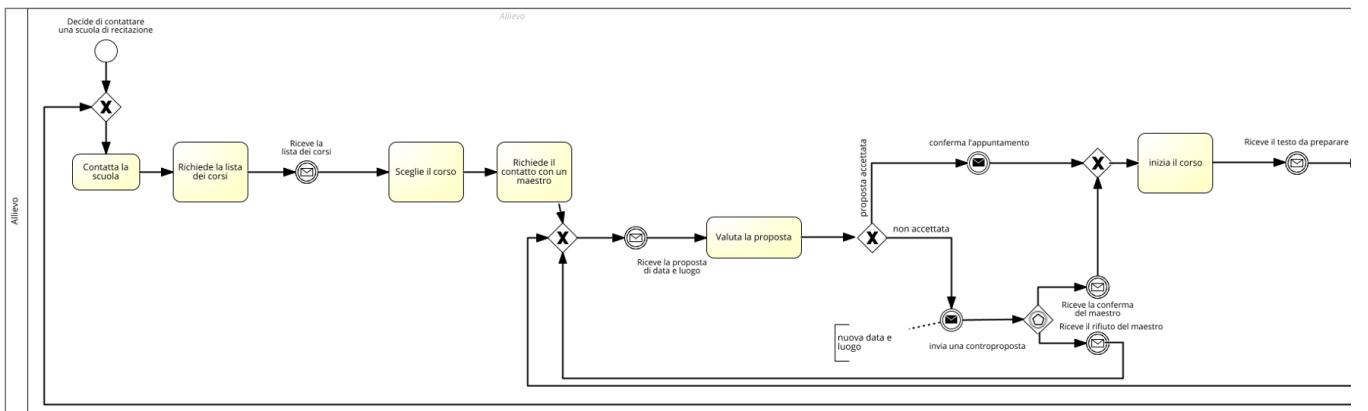


Figura 2.1: Pool Allievo inizio

Abbiamo assunto che l'allievo e il maestro si incontrino e che l'allievo *"Esegua la prova proposta dal maestro"* collegato con un **message flow arrows** che rappresenta l'esecuzione della prova e la successiva valutazione da parte del maestro. Qui infatti l'allievo aspetterà una valutazione. Potrà essere valutato o negativamente, di conseguenza il maestro gli dirà *"Ripeti l'esecuzione"*, oppure positivamente, gli comunicherà *"Esecuzione corretta"*. In questo caso il maestro proporrà di *"Eseguire la prova successiva"*, per il caso in cui ci siano più prove da ascoltare, oppure *"l'allievo ha terminato la lezione"*. Terminata la lezione, l'allievo riceverà dalla scuola i dettagli di pagamento per la lezione conclusa e si attiverà il task *"Pagamento elettronico"* che arriverà alla scuola con un **message flow arrows**. L'allievo potrà a questo punto prendere una decisione, o proseguire il corso, quindi avviserà la scuola e il flusso tornerà allo **XOR join** precedente alla recezione della proposta di data e luogo per lezione successiva, oppure deciderà di terminare il corso e lo comunicherà alla scuola.

Il testo inoltre richiede di modificare i processi in modo che al termine del corso lo studente possa scegliere di iniziare un nuovo percorso di studi. Abbiamo deciso di inserire direttamente questa possibilità aggiungendo uno **XOR split** in base alla sua decisione. Se sceglierà di iniziare un nuovo percorso di apprendimento, il flusso tornerà all'inizio del percorso dove contatterà la scuola e questa gli invierà la lista dei corsi. Abbiamo deciso di far tornare il processo in questo punto perché durante il tempo in cui si è svolto il corso precedente, possono essere stati attivati o disattivati dei corsi. Se invece deciderà di chiudere completamente i rapporti con la scuola, lo comunicherà a quest'ultima e il processo si concluderà.

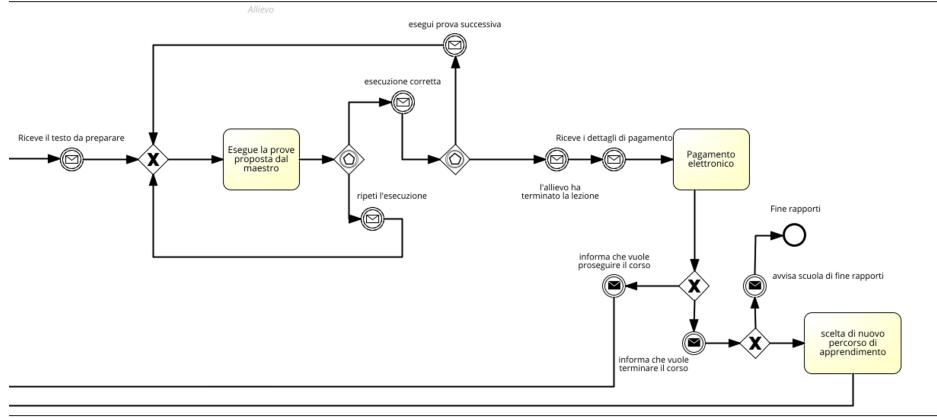


Figura 2.2: Pool Allievo fine

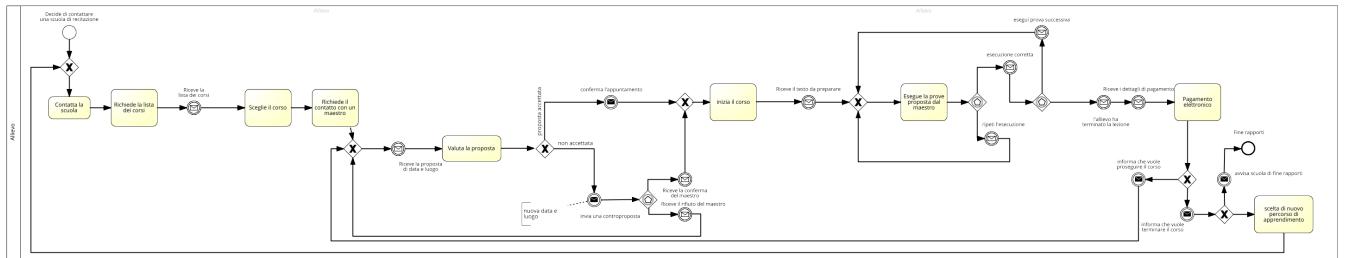


Figura 2.3: Pool Allievo completo

2.2 Pool Scuola di recitazione

Il pool della scuola, come detto in precedenza, si attiva appena riceve la richiesta di contatto dall'allievo, prima di questo era nello stato *"In attesa di contatto"* proprio perché abbiamo immaginato la segreteria della scuola che riceva questo genere di richieste da più studenti durante una giornata lavorativa. Una volta messi in contatto e ricevuta quella che è la richiesta della lista dei corsi disponibili, la scuola tramite un'attività *"Invia la lista dei corsi"* all'allievo, riceverà tramite un **message flow arrows** quella che sarà la scelta del corso e attiverà il pool del maestro inviandogli la richiesta dell'allievo di voler iniziare il corso. A questo punto i due attori principali entrano in contatto.

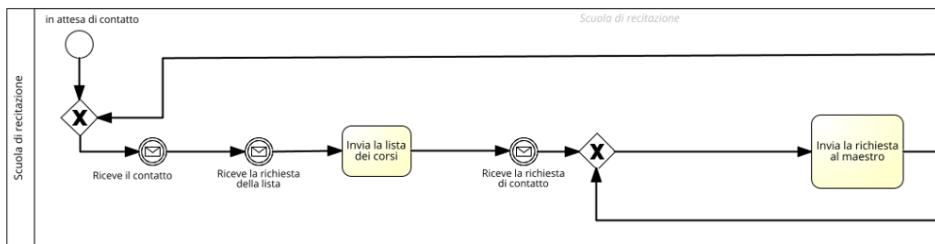


Figura 2.4: Pool Scuola inizio

Una volta che la lezione fra allievo e maestro ha avuto luogo, quest'ultimo contatterà la scuola e la informerà che la lezione è terminata. La scuola *"Invia dettagli di pagamento"* all'allievo tramite un **message flow arrows** e riceverà il pagamento da parte di questo. Una volta ricevuto entrerà in uno stato di attesa di evento, rappresentato dall'**event based gateways** in cui lo studente prenderà la decisione sul proseguimento o meno del corso. Se continuerà il corso, la scuola a sua volta, avviserà il maestro e ritornerà nello stato in cui i due attori principali si mettono in contatto. Se terminerà il corso, la scuola verrà informata di ciò e a sua volta avviserà il maestro. Data la modifica richiesta, a questo punto aggiungiamo la parte in cui lo studente può chiedere di frequentare un altro percorso, se ciò avviene, la scuola verrà prima di tutto informata, e successivamente, avviserà il maestro che ritornerà nello stato iniziale in cui aspetta di essere contattato dalla scuola. Abbiamo assunto per semplicità che il pool riguardi un unico generico maestro. Se l'allievo invece decide di chiudere completamente i rapporti, informerà la scuola che a sua volta avviserà il maestro e il processo si concluderà. Tutta la messaggistica viene gestita da dei **message flow arrows** tra i 3 attori.

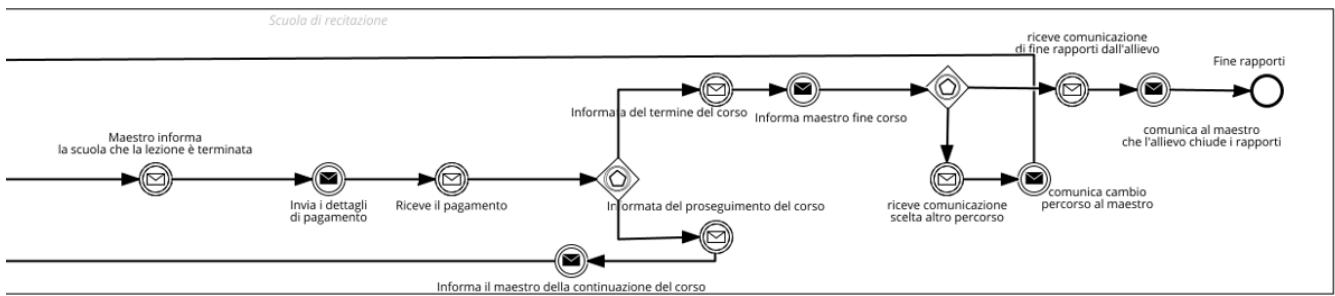


Figura 2.5: Pool Scuola fine

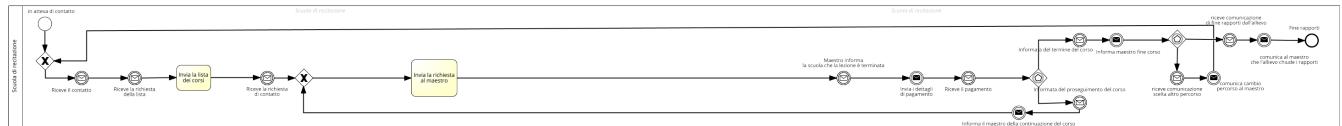


Figura 2.6: Pool Scuola completo

2.3 Pool Maestro

Il pool del maestro inizia con un "attesa di contatto" da parte della scuola di recitazione, la quale dovrà metterlo in comunicazione con l'allievo avvisandolo tramite un **message flow arrows**. Ricevuta la richiesta di questo, il maestro controllerà le sue disponibilità per fissare un appuntamento che sia consone con i suoi impegni. Effettuata la decisione, proporrà all'allievo una data e un luogo, che come specificato precedentemente. In questo momento il maestro si troverà davanti un **event based gateways**, in attesa di una decisione da parte del ragazzo/a che, come specificato in precedenza, potrà accettare o rifiutare. Il maestro potrà ricevere la conferma dell'appuntamento, oppure una controproposta. In questo ultimo caso, abbiamo deciso di effettuare una scelta attraverso un **XOR split** dove il maestro potrà accettare la proposta o comunicare il rifiuto. In entrambi i casi verrà avvisato l'allievo tramite un **message flow arrows**. Se comunica il rifiuto, tornerà nell'**XOR join** per ricontralare le sue disponibilità e continuare gli eventi del ciclo. Nel caso di accettazione dell'appuntamento inizierà il corso e "Invierà il testo da preparare" all'allievo tramite un **message flow arrows**. A questo punto i due attori si incontreranno per svolgere una lezione.

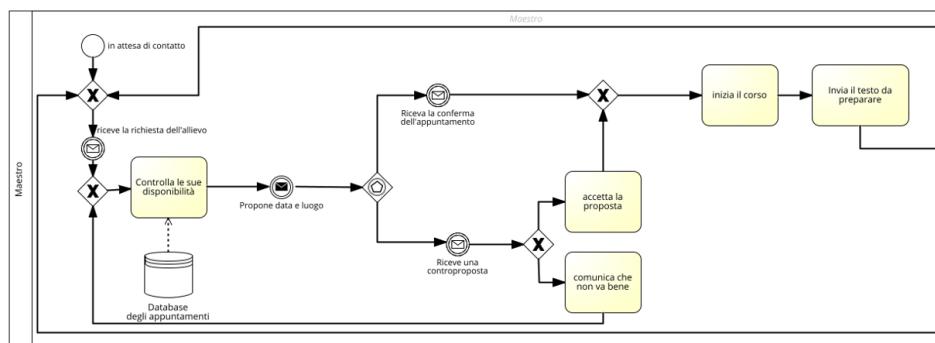


Figura 2.7: Pool Maestro inizio

Il problema propone di modellare uno scenario in cui l'allievo dovrà eseguire una serie di prove. Così abbiamo deciso di progettare un ciclo in cui il maestro "ascolta l'esecuzione della prova" e "valuta l'esecuzione". Se è corretta, lo comunica all'allievo ed, o propone di "eseguire una prova successiva" oppure "comunica il termine della lezione". Per eseguire la prova successiva, il maestro dovrà ascoltare di nuovo l'esecuzione, valutarla, fino a quando non sarà soddisfatto e informerà l'allievo della conclusione dell'insegnamento. Lo stesso processo viene ripetuto se per il maestro "l'esecuzione non è corretta". Al termine della lezione, il docente "informa la scuola della fine della lezione" attraverso il **message flow arrows**. Quindi il maestro viene messo in attesa, tramite un **event based gateway** di informazioni da parte della scuola di recitazione. Abbiamo deciso che se questa invierà al maestro che l'allievo vorrà continuare il corso, allora egli dovrà riattendere il contatto dello studente e ricominciare tutto il processo. Sennò la scuola comunicherà al docente che l'allievo vorrà terminare il corso.

Di conseguenza il maestro potrà o "ricevere informazioni sulla chiusura dei rapporti dell'allievo con la scuola" da parte di essa, e quindi finire il processo, oppure ricevere (sempre dalla scuola) una "comunicazione della scelta dell'allievo di intraprendere un altro percorso". Anche in questo ultimo caso il maestro tornerà all'inizio del processo e ripeterà i passaggi consecutivi.

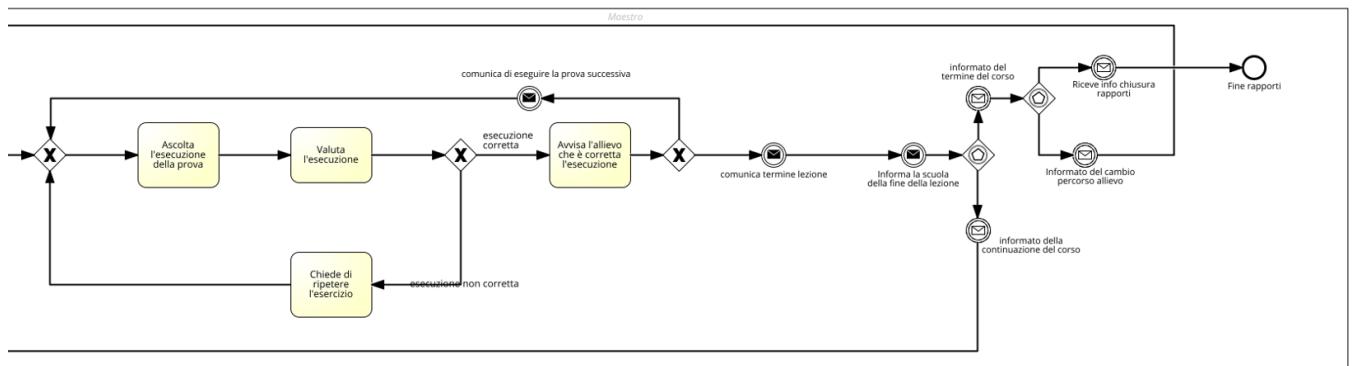


Figura 2.8: Pool Maestro finale

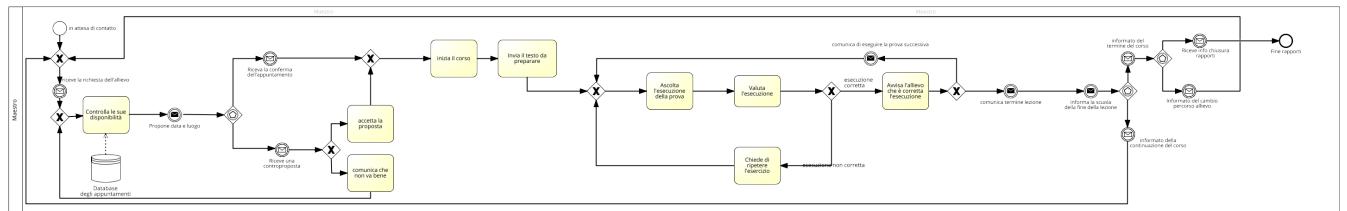


Figura 2.9: Pool Maestro completo

Nella pagina successiva mostriamo il **BPMN diagram** completo con la notazione adeguata e i **message flow arrows** non presenti precedentemente.

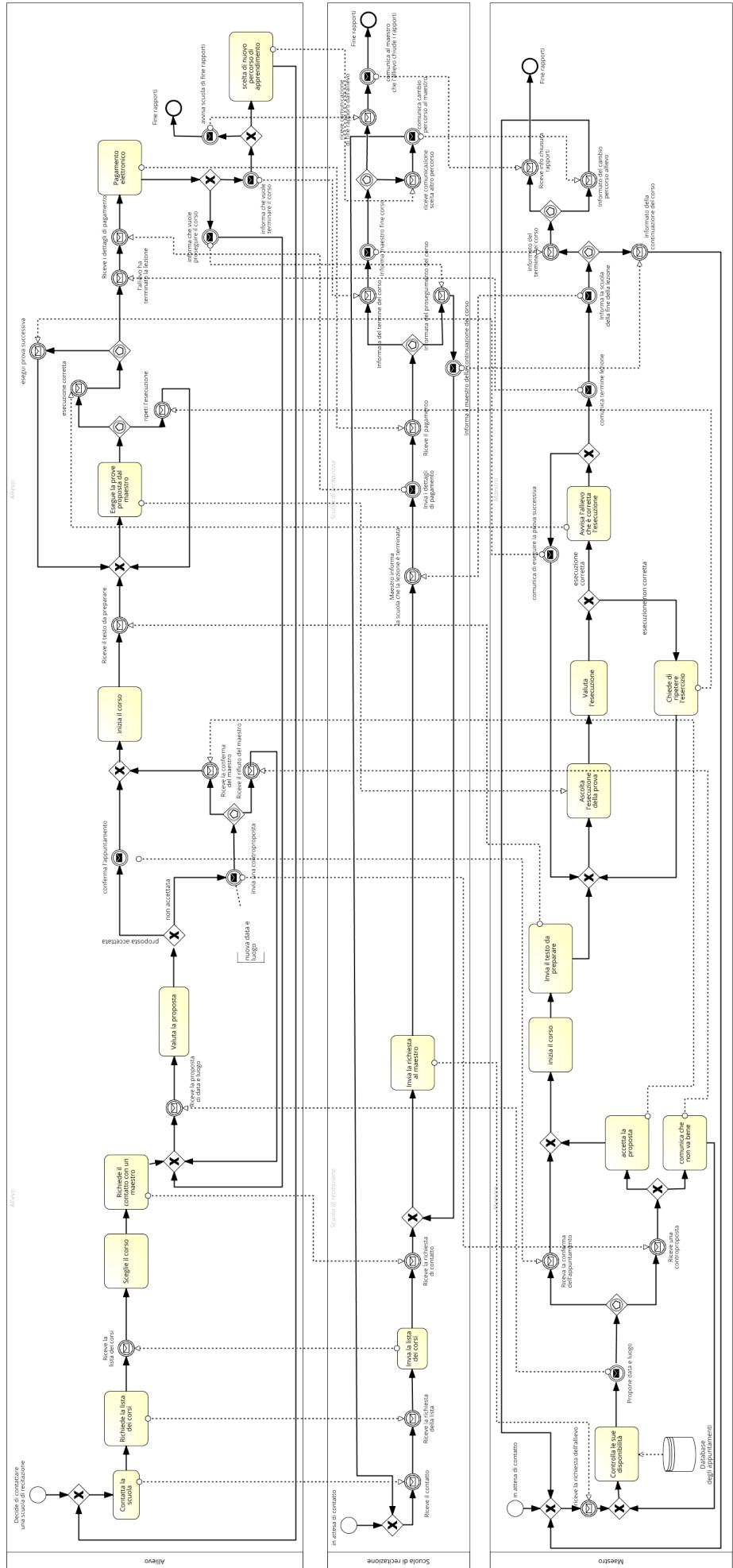


Figura 2.10: BPMN completo

3 PETRI NETS ANALISI

I **BPMN diagrams** sono stati successivamente tradotti in dei **workflow nets** di modo che fosse possibile effettuare l'analisi semantica del processo. Il processo di traduzione è stato sviluppato in tre fasi e vale per ognuno dei nostri nets:

1. trasformazione delle attività e degli eventi in transizioni.
2. aggiunta dei vari gateways come place.
3. inserimento del place iniziale e finale

Inoltre è possibile osservare che, dato che ognuno dei nostri processi è *finito*, i *coverability graphs* coincidono con i *reachability graphs*.

3.1 Allievo

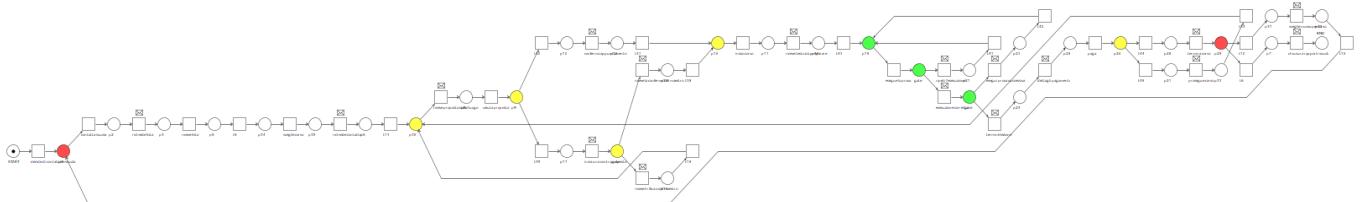
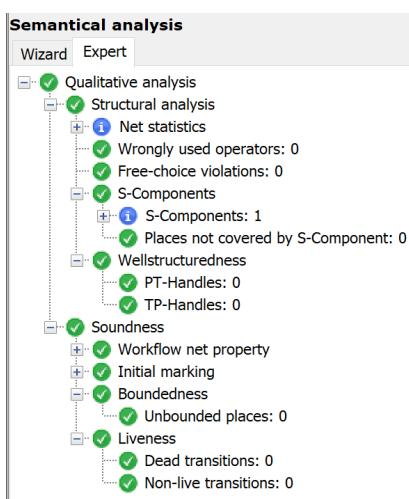


Figura 3.1: Workflow nets Allievo

Structural e sound analysis:



- E' un workflow nets perchè ha :
 - place iniziale = \emptyset
 - place finale $\bullet = \emptyset$
 - ed esiste un percorso dal place iniziale al finale
- N è un **S-net** quindi è **sound**
- Deadlock free, bounded, free-choice e strongly connected
- Well-structured : non ci sono **PT/TP-handles**
- S-coverable da 1 S-components che copre 77 elementi del net

Il net ha:

- **36 places**
- **41 transitions**
- **82 archi**

Il *coverability graph* ha **36 vertici** e **41 archi**.

Figura 3.2: Semantical Analysis
del workflow net dell'allievo.

3.2 Scuola di recitazione

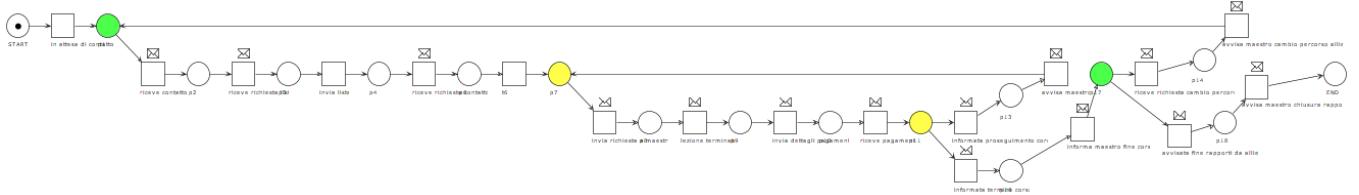
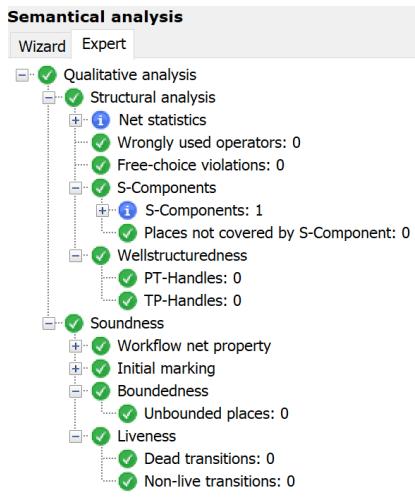


Figura 3.3: Workflow nets Scuola

Structural e sound analysis:



- E' un workflow nets
- E' Free choice
- N è un **S**-net quindi è **sound** e **safe**
- Deadlock free, bounded e strongly connected
- Well-structured : non ci sono **PT/TP-handles**
- S-coverable da 1 S-components che copre 35 elementi del net

Il net ha:

- 17 places
- 18 transitions
- 36 archi

Il *coverability graph* ha 17 vertici e 18 archi.

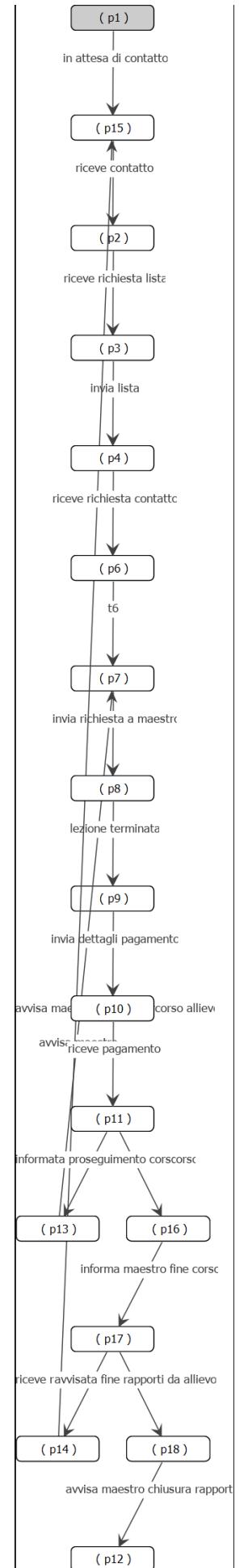
Figura 3.4: Semantical Analysis del workflow net della scuola di recitazione.

Abbiamo deciso di inserire il coverability graph per la Scuola, in quanto è la più semplice da analizzare.

Il *coverability graph* coincide con il *reachability graph*. Possiamo osservare che in quanto finito, è bounded. Invece per quanto riguarda la liveness, questa non sarà soddisfatta perché se attiviamo "*in attesa di contatto*", p1 diventa morta. Invece se analizziamo il grafo di N^* tutte le proprietà sono soddisfatte. C'è un ciclo che parte da p7, scende fino a p11 e poi ritorna se si attiva il token in p13. Il secondo ciclo parte da p15 e scende lungo il grafo, fin a quando non arriva a p17 dove dovrà prendere una scelta. Se si attiva la transizione "*avvisata cambio percorso allievo*", si ritorna a p15, se no il processo termina attivando la transizione "*avvisata chiusura rapporti*". In questo caso un token entra nel place p12 e il processo è completo.

Questo grado mostra anche che la **option to complete** e la **proper completion** sono soddisfatte. Una volta che un token entra in p12 non ci sono altre token isolati. Inoltre, l'unico modo per tornare al place iniziale p1 è aggiungendo la *reset transition*. Ovviamenete abbiamo anche **no dead task** dato che per ogni transizione ha almeno un arco nel coverability graph.

Quindi N è **sound** e **safe** perchè è 1-bounded.



3.3 Maestro

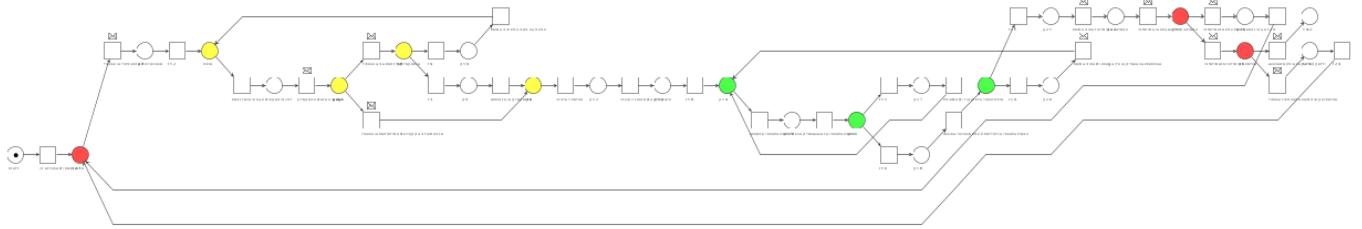


Figura 3.5: Workflow nets Maestro

Structural e sound analysis:

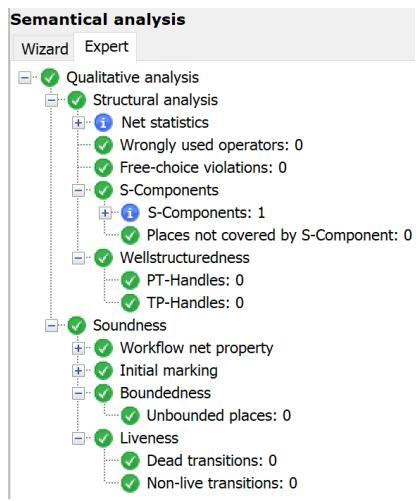
- E' un workflow nets
- E' Free choice
- N è un **S-net** quindi è **sound** e **safe**
- Deadlock free, bounded e strongly connected
- Well-structured : non ci sono **PT/TP-handles**
- S-coverable da 1 S-components che copre 57 elementi del net

Il net ha:

- **26 places**
- **31 transitions**
- **62 archi**

Il *coverability graph* ha **26 vertici** e **31 archi**.

Figura 3.6: Semantical Analysis
del workflow net del maestro.



4 Conclusioni

Come ultimo step abbiamo aggiunto, ai singoli Petri net degli attori, i quali comunicano attraverso i **message flow arrows** nella modellazione BPMN, ulteriori places tra le interfacce per permettere loro di scambiare messaggi. In conclusione si ha un **workflow modules** strutturalmente compatibile con quella dei singoli attori con l'aggiunta dei places. Alla fine abbiamo unito i places con le diverse transizioni di scambio messaggio per rendere l'intero sistema **sound**. Inizialmente abbiamo notato che nella progettazione iniziale del workflow net la proprietà della **proper completion** non veniva rispettata. Analizzando la rete abbiamo notato che la causa di questo problema fosse dovuta ad un mancato scambio di messaggi tra il maestro e la scuola. Integrati quest'ultimi la rete è risultata **sound**, quindi tutte le proprietà sono state rispettate.

Di seguito vi è una rappresentazione del *workflow system* creato con WoPed e la relativa analisi di Woflan. Nella pagina successiva mostriamo lo stesso processo ma utilizzando un design più leggibile ottenuto con la funzione *Optimize layout* di Woped. Infine, nella pagina conclusiva rappresentiamo il **Coverability graph** del nostro workflow system.

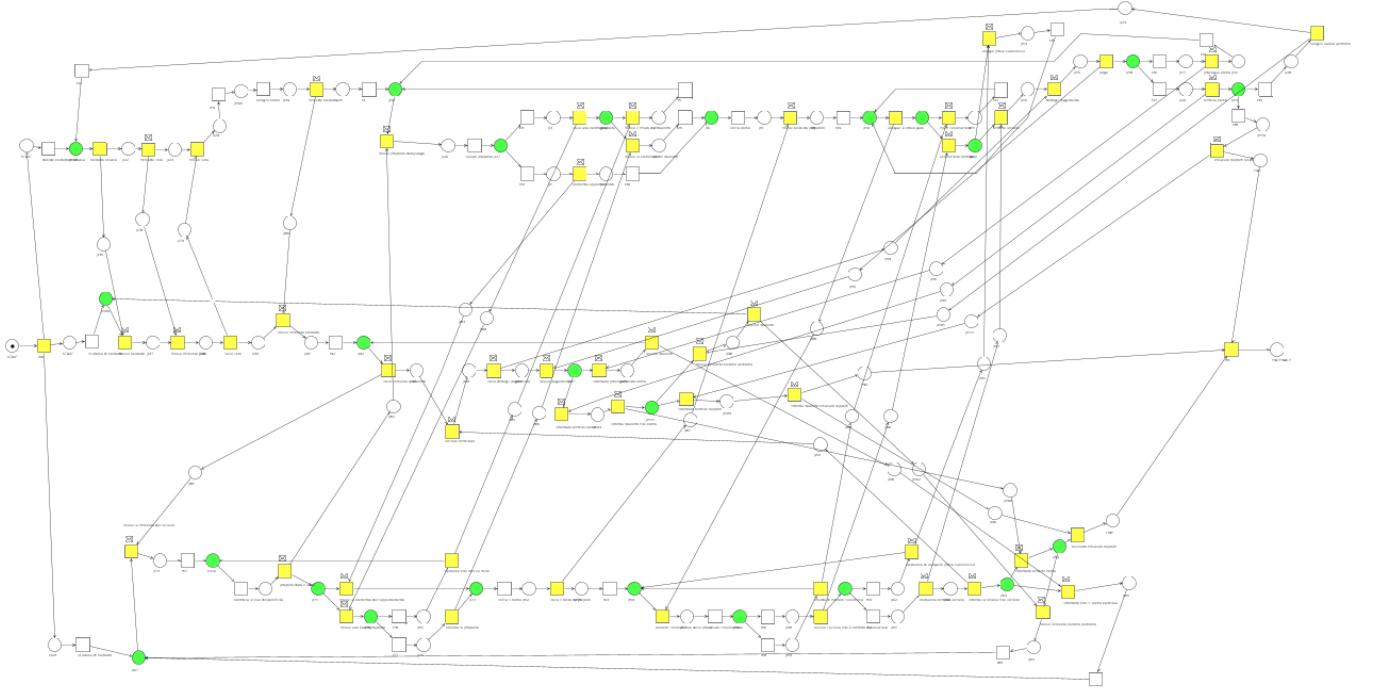


Figura 4.1: Perti-nets Scuola di recitazione (del progetto completo)

Il Net ha:

Diagnosis	<ul style="list-style-type: none"> ✓ The process definition is a workflow process definition ✓ All conditions are proper ✓ All conditions are covered by threads of control ✓ All tasks are live ✓ The workflow process definition is sound
Properties	<ul style="list-style-type: none"> ✓ Process definition <ul style="list-style-type: none"> ✓ Conditions: 108 ✓ Tasks: 92 ✓ Workflow process definition <ul style="list-style-type: none"> ✓ Start conditions: 1 ✓ End conditions: 1 ✓ Useless tasks and conditions: 0 Condition properness <ul style="list-style-type: none"> ✓ Threads of control: 877 Task liveness <ul style="list-style-type: none"> ✓ Dead tasks: 0 ✓ Non-live tasks: 0

Figura 4.2: Report Woflan Scuola di recitazione (del progetto completo)

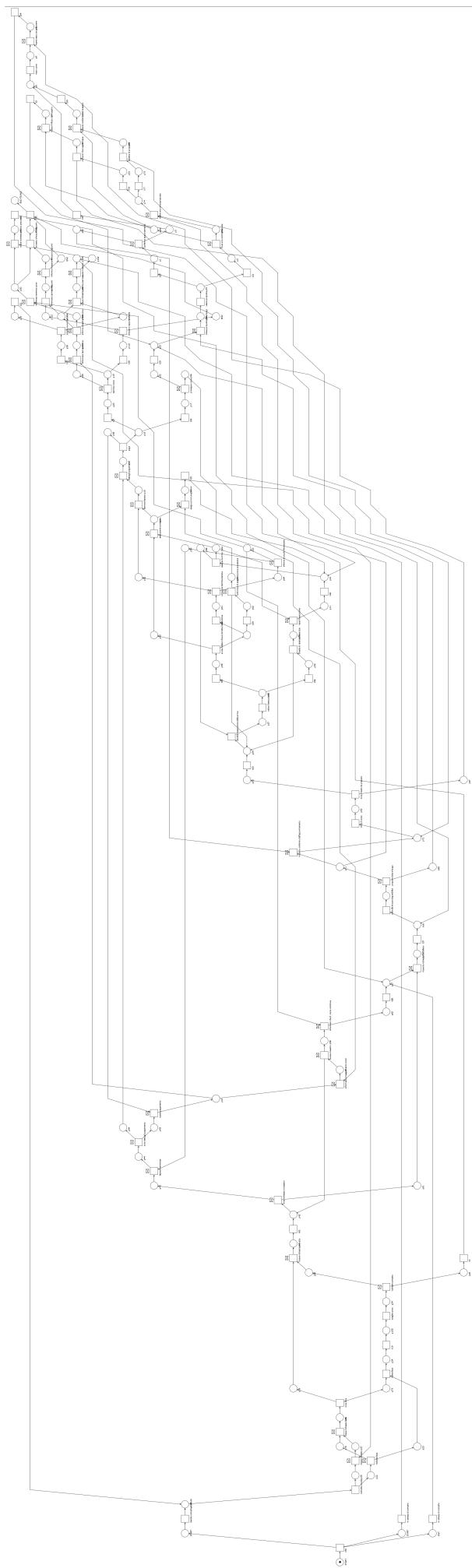


Figura 4.3: Petri-nets ottimizzato

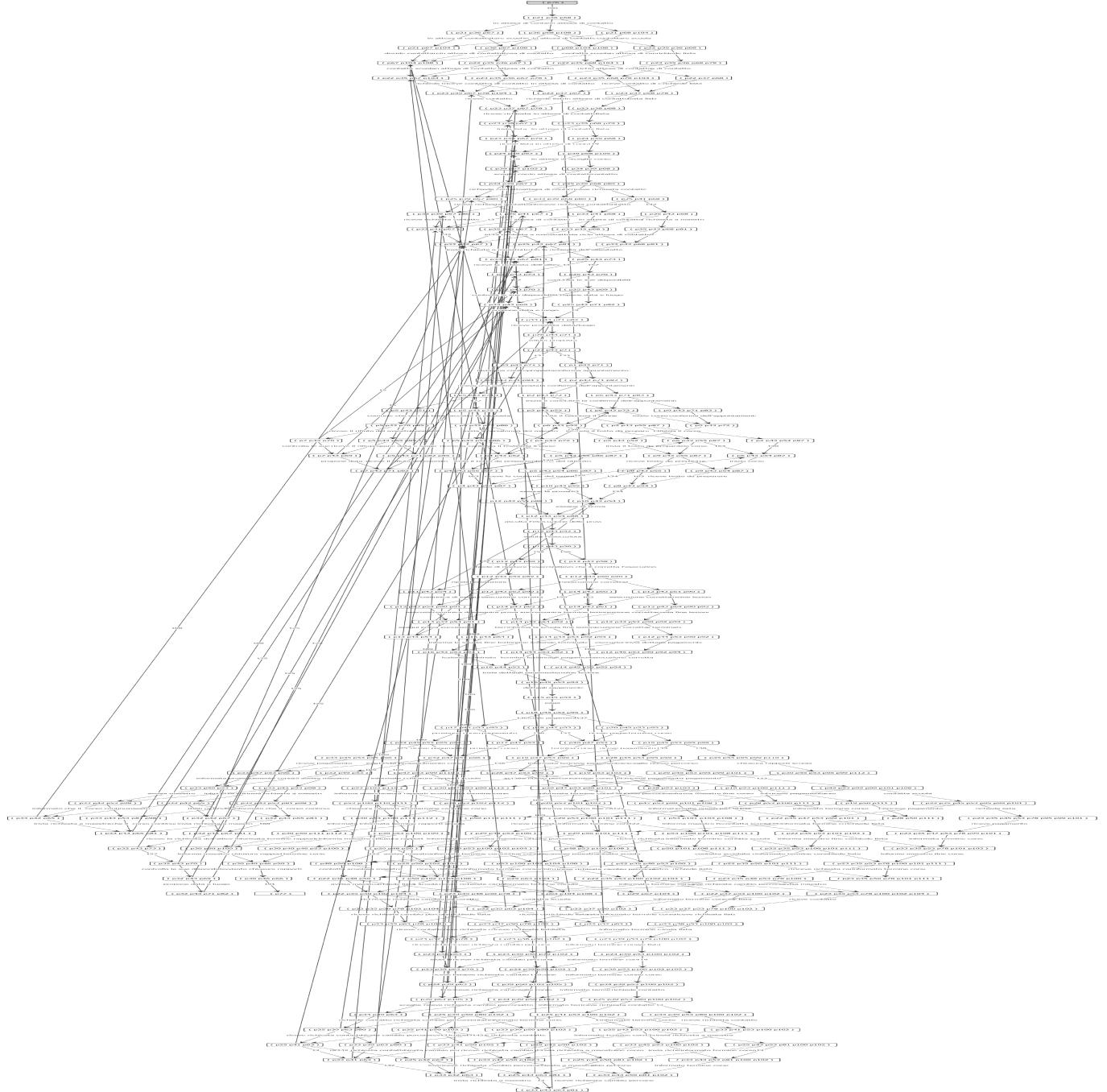


Figura 4.4: Coverability Graph Scuola di recitazione